EROTETIC LOGIC AS A SPECIFICATION LANGUAGE FOR DATABASE QUERIES

by

GARY JAMES JASON

Ph.D., University of Illinois, 1982

A MASTER'S THESIS

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1986

Approved by:

[illegible]
Major Professor

Contents

Page

Chapter 1: Introduction: An Overview Of Fourth Generation Logics 1

Chapter 2: A Formal Erotetic System 11

Chapter 3: Formal Queries 26

Chapter 4: Conclusion 30

Notes 32

Bibliography ....

1) Introduction: An Overview of Fourth Generation Logics

In fairly recent times, logicians have developed a profusion of non-classical logics: many-valued, modal, temporal, fuzzy and so on. Some of these logics, which I shall call "fourth generation logics," have already been developed and exploited by computer scientists. Temporal logic has been used in AI and for the formal specification and verification of concurrent programs. Many-valued logic has been used in circuit design, and fuzzy logic in AI. In this paper I would like to discuss an area of fourth generation logic, erotetic logic, that may well be of use to computer scientists working AI and database theory.

I have coined the term "fourth generation logics" for two reasons. First, while the more customary term for these logics is "deviant logics," that phrase makes it sound as if people who do research on such logics are somehow strange or weird. Second, the phrase "fourth generation" is at least roughly historically accurate.

We can distinguish generations of logic, in part on the basis of time of origin. Following Bartley [3] and Jørgensen [10], we can view the first generation of logic as beginning with Aristotle in the fourth century B.C.. First generation logic, "traditional" ("Aristotelian" or "syllogistic") logic, focused upon reasoning in natural language. Logicians were concerned to formulate rules for casting propositions into categorical forms, and for assessing arguments ("syllogisms") built up out of those categorical propositions.

The second generation of logic, Boolean or algebraic logic, dates from the work of George Boole (1847) and the work of DeMorgan, Venn, Lewis Carroll, and others later in the nineteenth century. The second generation logicians kept the earlier concern with natural language argumentation, but extended logic to cover a wider class of arguments (ones involving relational propositions). More importantly, the algebraic logicians subtly shifted the focus of research to the application of mathematics to logic. Graphical methods (Venn diagrams, Euler diagrams Carroll diagrams and so on) for assessing syllogisms were developed, and methods for expressing propositions algebraically were devised.

Thus the second generation logicians greatly expanded on the scope of analysis beyond traditional syllogistic, and greatly increased the formalization of logic. They also were concerned with devising techniques for _drawing_ relevant inferences from premises, i.e., with _solving_ problems or _discovering_ solutions, as opposed to merely verifying the correctness of already formulated proofs. This concern shows up in the exercises found in second generation logic texts (such as Lewis Carroll's _Game of Logic_): puzzles that call upon the student to determine what information about some situation can be derived from given information. (Vestiges of the second generation concern over logical puzzles are still found in some elementary logic texts, and several modern logicians--most notably E.R. Emmet and Raymond Smullyan--have written books of puzzles.)

Third generation logic is classical First Order Logic (hereafter "FOL"). While we might date the birth of FOL with the publication of Frege's _Begriffschrift_ in 1879, it is more accurate to date the third generation in logic to the first decade of this century, when Russell brought Frege's work to public notice. Eminent logicians such as Peirce, Whitehead, Russell, Gödel, Church, Gentzen and many others rapidly developed FOL, with many major results in place by the 1930's. Third generation logic is concerned much less with argumentation in natural language than with the foundations of mathematics and proof theory. Hence the focus of research was on consistency proofs, axiomatization, decision procedures and so on. Indeed, many third generation logicians were avowed logicists, i.e., they wanted to reduce all of mathematics to logic and set theory.

Fourth generation logic is harder to date. We might date it late in the nineteenth century, with Peirce's work on many-valued logic. It would be more accurate, however, to fix the date in the 1920's, and 1930's, with the work of Lukasiewitz on three-valued logic and the work of C.I. Lewis on strict entailment (modal logic). But in terms of purpose, the difference between third and fourth generation is clear. Fourth generation logicians are much less concerned with the foundations of mathematics, and are once again concerend with reasoning and knowledge expressed in natural language. Typically, the fourth generation logician feels that FOL is inadequate to explicate some interesting feature of rational discourse, and he seeks to find some modification of or replacement for FOL. The fourth generation logician has the same goal of the first and second generation logicians, but wants to use some formal apparatus in addition to (or in lieu of) FOL.

That FOL does not sufficiently explicate all natural language reasoning is by now conceded by most logicians. (Indeed, some logicians have gone to the extreme of arguing that FOL should not be taught as "logic"--see, for example [18] pp. 447-456.) This is perhaps not surprising: FOL explicates mathematical reasoning well, but in mathematics there is no cause and effect, before and after, or questioner and respondent.

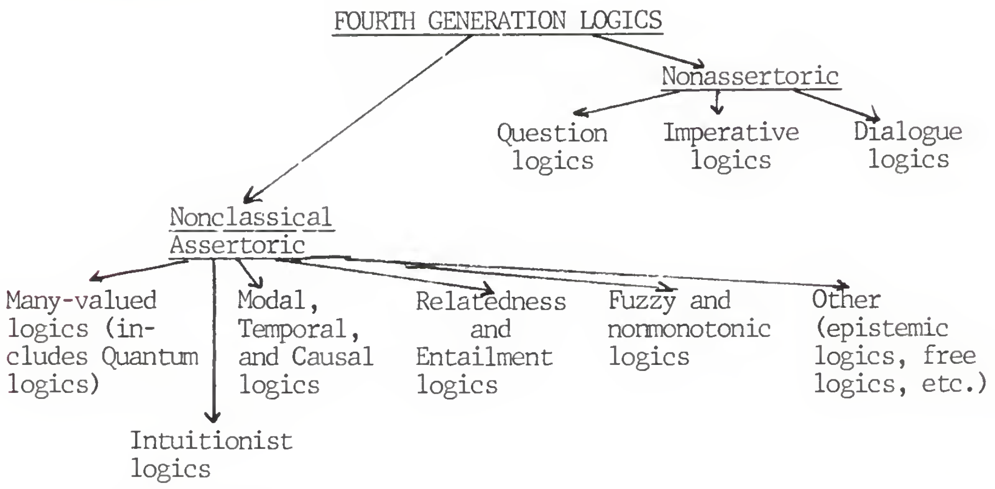
In any event, in the last forty years there has been a proliferation of fourth generation logics. Even classifying them is controversial. Some (such as Turner [20] and Haack [8]) have divided fourth generation logics into those which seek to extend FOL and those which seek to replace it. For example, standard modal logic extends FOL by adding modal operators, but the theorems of FOL remain valid. On the other hand, many-valued and intuitionist logics replace FOL in the sense that various theorems of FOL (e.g., the Law of Excluded Middle) fail in those logics.

I find that distinction unhelpful. To begin with, modal logics are many-valued in that they add a third truth value (necessary truth) to the classical values (true and false). More generally, logics don't seek to do anything: logicians do. And most fourth generation logicians don't seek to replace FOL; like scientists in other fields, they view conservationism as a virtue in a theory. That is, faced with two new proposed theories which aim to succeed where an established theory has failed scientists will naturally choose the one that least conflicts with the established theory (all things being equal). A fourth generation logician doesn't seek to replace FOL--he may be driven to it, but he doesn't seek it.

It may be suggested that I have misconstrued the proposed division.

The point is (it might be urged) that some alternate logics accept as theorems the theorems of FOL, and some do not. But this seems to be an unhelpful point. Many of those logics in which some classical theorems fail have analogues which replace the failed theorems--for example, while the Law of Excluded Middle fails in a three-valued logic, the Law of Excluded Fourth may still hold.

I suggest, then, that a more interesting division of fourth generation logic is between assertoric logics (those that concern logical connections between statements) and nonassertoric logics (those concerned with questions, commands, and such like). Some of the major types of fourth generation logic are listed below.


FOURTH GENERATION LOGICS
Nonassertoric
Question logics
Imperative logics
Dialogue logics
Nonclassical Assertoric
Many-valued logics (includes Quantum logics)
Intuitionist logics
Modal, Temporal, and Causal logics
Relatedness and Entailment logics
Fuzzy and nonmonotonic logics
Other (epistemic logics, free logics, etc.)

Several references are worth noting. [7] has a good survey of fourth generation logics and the issues surrounding them from the philosopher's point of view, while [19] has a survey of temporal, many-valued and fuzzy logic from the AI point of view. The classic survey of modal logic is [7] (updated by [8]). A fine survey of many-valued logic is [15] and of temporal logic is [16]. AI researchers are currently exploring the potential of relevance logic, and the Bible of such logics is [1].[1]

A fourth generation logic that is unfamiliar to many computer scientists is erotetic logic.

Roughly put, erotetic logic is the logic of questions. More precisely, if standard logic (assertoric logic) involves explicating statements and the relations between statements, erotetic logic is the extension of assertoric logic to explicate questions and the relations between questions and answers. By "explicate questions" I mean reveal the structure of questions in a way analogous to the way assertoric logic explicates the structure of statements. For example, in standard FOL the two relational sentences "John loves Mary" and "Mary is loved by John" are symbolized the same way ("Loves (John, Mary)") because while the surface structures of the English declaratives differ, the semantic structures of the statements are the same. Similarly, an adequate erotetic logic should symbolize

(1a) What are the good restaurants in Topeka?

(1b) Which Topeka restaurants are good?

the same, because while the surface structures of the English interrogatives differ, the semantic structures of the questions are the same.

Again, just as an assertoric logic categorizes statements (as being conditionals, negations, existentially quantified or whatever), an adequate erotetic logic should be able to categorize questions, and reveal the differences between:

(1a) What are the good restaurants in Topeka?

(2) What are some good restaurants in Topeka?

(3) What is a good restaurant in Topeka?

(4) Why are there no good restaurants in Topeka?

(5) Why do you say that there are no good restaurants in Topeka?

(6) Is The Loft a good Topeka restaurant?

(7) Where is The Loft located?

By "explicate the relation between questions and answers" I mean the formulation of rules governing the acceptability of a given answer to a given question. For example, question (1a) admits the answer "The Loft and the Amarillo Grill are the only good restaurants in Topeka," while none of the others do. Crucial here is the notion of a <u>direct</u> answer, an answer which supplies the requested information, but no more. That is why "The Loft and the Amarillo Grill are the only good restaurants in Topeka" is not an acceptable answer to (3), because (3) only requests one example of a good restaurant in Topeka, not a complete list.

Thus to construct an adequate erotetic logic we must do three things. First, we must decide upon a typology of questions. Second, we must develop a formal system--presumably an extension of some formal assertoric logic--which will allow us to symbolize questions. Third, we must formulate a set of admission rules, i.e., rules that specify the syntactic form of an acceptable answer to a given type of question.

Developing a typology of questions requires considerable work--mainly of a philosophic and linguistic nature. There is a large body of literature on this topic (see the bibliography to [5]). To simplify the issues treated in this paper, we will adopt the following typology:

$$\begin{tikzcd}
 & & \underline{\text{QUESTIONS}} \arrow[dl] \arrow[drr] & & \\
 & \underline{\text{Simple}} \arrow[dl] \arrow[d] \arrow[dr] \arrow[drr] & & & \underline{\text{Compound}} \\
\text{Whether} & \text{Which} & \text{Why} & \text{How} &
\end{tikzcd}$$

Whether-questions: These questions present their subjects (their range of alternatives) explicitly and request the respondent to select from among them. Example:

(8) Is Fred rich?

(9) Which should Fred take: FORTRAN, COBOL, Pascal or BASIC?

(8) presents the alternatives "Fred is rich" and "Fred is not rich," while (9) presents the alternatives "Fred should take FORTRAN," "Fred should take COBOL," "Fred should take Pascal," and "Fred should take BASIC."

Which-questions: These questions present their subjects by means of statement functions, and request the respondent to select the data satisfying those functions. Examples:

(10) Which cars are expensive?

(11) Who will come to the party?

(12) What are Fred's office hours?

(10) presents the subject by means of the proposition function "x is a car and x is expensive," (11) by "x is a person and x will come to the party, and (12) by "x is an office hour of Fred."

Why-questions: These questions are of two sorts. The first sort of why-question is a request of the respondent to explain an acknowledged fact. Examples:

(13) What is the cause of the oversupply of Ph.D.'s?

(14) Why do dogs bark?

The second sort of why-question is a request of the respondent to prove a point that the questioner doubts. Examples:

(15) Why do you say that Fred is unhappy?

(16) What are his reasons for buying that stock?

The topic of why-questions is important in philosophy of science, and will probably become important in AI; however, it is not relevant to this paper.

<u>How-questions</u>: These questions also are of two sorts. The first sort request of the respondent a set of instructions. Examples:

(17) How do I get to Oklahoma City?

(18) How can we buy a new car?

The second sort of how question is a request for the respondent to impart a skill. Examples:

(19) Can you show me the fox-trot?

(20) Do you know how to get this bottle-cap off?

Again, while the topic of how-questions is interesting and potentially relevant to AI, it is not relevant here.

<u>Compound questions</u>: Questions, like statements, can be compound. Examples:

(21) Who is that masked man, and what does he want?

(22) Where and when shall we get together?

Questions can be compounded with statements:

(23) If Fred wants to buy a car, where will he get the money?

In this paper we will not pursue the topic of compound questions.[2] Questions that are truth-functionally compounded together we can handle in an obvious way. For example, an acceptable answer to (22) would simply be a conjunction of acceptable answers to "Where will we get together?" and "When will we get together?" Moreover, most cases in which statements are compounded with questions seem to be cases in which the statement is a guard against presupposition failure. For example, we would ask (23) rather than the simple question

(24) Where will Fred get the money to buy the car he wants?

because (24) presupposes that Fred wants to buy a car, and we may not be sure that he does. But presupposition failure will be accounted for in our treatment (below) of simple questions.

In short, the formal apparatus we develop will be geared toward which-and whether-questions, as they seem to be the most crucial for database theory (as opposed to AI or philosophy of science).

2) A Formal Erotetic System

Having picked out the types of questions in which we are interested, we now construct a formal system for putting those questions.[3] Since a formal erotetic language is an estension of an assertoric language, we need first to lay out the grammar of that underlying logic. The language we will take to be FOL. The vocabulary (or "alphabet") consists of:[4]

infinitely many variables: x, y, z, x_1, y_1, z_1,...

individual constants: a, b, c,...

predicates of any arity: A, B, C,...

punctuation symbols: (,), [,], {, }

logical constants: $\rightarrow$(if then), & (and), $\forall$(universal quantifier), $\exists$ (existential quantifier), v (or), ~ (not), $\leftrightarrow$(if and only if)

In philosophic literature, predicates and constants are kept to one letter. In computer science literature, predicates are often whole words (e. g. "RESTAURANT (x)"). The difference is purely one of style, but for readability we prefer the computer science style.

The well-formed formulas are characterized as follows. We say that variables and constants are terms. We say that an n-ary predicate followed by n terms is an atomic formula $P(t_1,\ldots,t_n)$. If all the t_i's are constants (i.e., individual names), then $P(t_1,\ldots,t_n)$ is a ground atomic formula (a "particular statement"). Then the set of well-formed formulas (wffs) is the set defined by the rules:

(1) any atomic formula is a wff;

(2) if p is a wff, then so is $\sim p$;

(3) if p and q are wffs, then so are p v q, $p\rightarrow q$, $p\leftrightarrow q$, and p & q;

(4) is v is a variable and p a wff, then so is $(\forall v)p$ and $(\exists v)p$;

(5) no other expressions are wffs.

We don't need a full-blown first order language to formulate a database. We say that a first-order language is a relational language if and only if it has at least one but at most finitely many constants, it has only finitely many predicates, has equality (denoted by '$x = y$') as a predicate, and has a set of "types." A <u>type</u> is a unary predicate t (a <u>simple</u> type) or boolean combinations of simple type. (Types allow us to capture the notion of the "domain" of a given relation).[5]

When types are defined we allow ourselves the luxury of type-restricted quantifiers. We will use

$(\forall x/_{T_1})\ \mathcal{A}x$

to mean

$(\forall x)\quad (T_1 x \rightarrow \mathcal{A}x)$

and

$(\exists x/_{T_1})\ \mathcal{A}x$

to mean

$(\exists x)\quad (T_1 x\ \&\ \mathcal{A}x)$

In a standard way we can interpret the language L. An interpretation M consists of a domain D of individuals over which the variables range, a mapping f of the constants of the language onto individuals of D, and a mapping E of the predicates of the language onto sets of tuples of individuals of the domain. We can then characterize truth or satisfiability under M (written $\models_M$) as follows. For a given mapping v of variables of L into D, define another mapping r as follows:

$r(c) = f(c)$ for every constant of L

$r(x) = v(x)$ for every variable of L

Then define $\models_{r,M}$ as follows:

1) $\models_{r,M} P(t_1,\ldots,t_n)$ iff $(r(t_1),\ldots,r(t_n)) \in E(P)$ for every atomic formula of L.

2) $\models_{r,M} A_1 \& A_2$ iff $\models_{r,M} A_1$ and $\models_{r,M} A_2$

3) $\models_{r,M} A_1 \vee A_2$ iff $\models_{r,M} A_1$ or $\models_{r,M} A_2$

4) $\models_{r,M} \sim A$ iff not $\models_{r,M} A$

5) $\models_{r,M} (A_1 \rightarrow A_2)$ iff $\models_{r,M} (\sim A_1 \vee A_2)$

6) $\models_{r,M} (A_1 \leftrightarrow A_2$ iff $\models_{r,M} ((A_1 \rightarrow A_2) \& (A_2 \rightarrow A_1))$

7) $\models_{r,M} (\forall x)\ A$ iff for all $d \in D$ $\models_{r^* M} A$ where r^* is identical to r except for mapping x to d

8) $\models_{r,M} (\exists x)\ A$ iff $\models_{r,M} \sim(\forall x) \sim A$

Then we can define $\models_M A$ as: $\models_M A$ (read "A is true in interpretation M") iff $\models_{r,M} A$ for all r. M is said to be a model for A if $\models_M A$.

In interpreting a relational language, we stipulate that equality hold of every individual in the domain, and we also stipulate a set of integrity constraints. An integrity constraint for a predicate P has the form (where $T_1,\ldots,T_n$ are the types):

$$(\forall x_1)\ldots(\forall x_n) \left[P\ (x_1,\ldots x_n) \rightarrow (T_1 x_1\ \&\ \ldots\ \&\ T_n x_n)\right]$$

For example, we might lay down a constraint upon

Wife (x,y)

that x be a woman and y be a man:

$$(\forall x)\ (\forall y)\ \left[\text{Wife}\ (x,y) \rightarrow (\text{Woman}(x)\ \&\ \text{Man}(y))\right]$$

Now that we have the assertoric logic set forth, we turn to the erotetic extension. Our goal to develop a formalism that will allow us to specify what it is for a given statement to correctly answer a given question. We will do this in two stages: first, we will devise a set of rules which govern the acceptability of answers, and then (in the next section) we will state a criterion for truth of answers.

Following Belnap ([4] and [5]), whose term system we will adopt, we view a question as having the form ?RS, where R is the request and S the subject. Roughly defined, the request specifies the amount of information desired, while the subject specifies the sort of information to be examined. The question mark we view as an operator which takes request-subject pairs onto questions. A query is just a question applied to (or "raised within") a database. In this section we will focus on representing questions without concern for databases; we will look at queries later.

We discuss the subject S first. The subject of a whether-question presents its range of alternatives explicitly. That is, the subject has the form of a set of statements

$A_1, A_2, \ldots\ A_n$

where none of the Ai's are repetitions or conjunctions solely of the others of the set. For example, the subject of the question

Is John married?

is the pair

$\{$John is married, John is not married$\}$

symbolically

$\{$married (John), $\sim$ married (John)$\}$

Again, the subject of

Has John stopped beating his wife?

is

$\{$Has_beaten_wife (John) & Now_beats_wife (John), Has_beaten-wife

(John) & $\sim$ Now_beats_wife (John)$\}$

A more elaborate example:

Which are on sale today: ham, turkey, or steak?

has the subject

$\{$on_sale (ham, today), on_sale (turkey, today), on_sale (Steak, today)$\}$

The subject of a which-question presents a potentially infinite set of alternatives, bymeans of statement functions, some of which are type constraints and the others the matrix. For example, the question

Which Topeka restaurants take credit cards?

has as subject matrix

accepts (x, credit_cards) & located (x, Topeka)

and as type constraint

restaurant (x).

We will separate the subject types from the matrix by a souble slash:

$\{$restaurant (x)//accepts (x, credit_cards) & located (x, Topeka)$\}$

In general we can symbolize the subject of a which-question as

$$\{T_1x,\ldots,T_rx_r // A(x_1,\ldots,x_n)\}$$

where $0 \leq r \leq n$.

Note two things. First, we can push the type constraints over into the matrix if we choose, although there is no good reason for doing so. For example, the subject

$\{$Woman(x), Man(y)//Wife(x,y)$\}$

can be put equivalently

Woman(x)//Man(y) & Wife (x,y)

or even

$\{\epsilon$ //Woman(x) & Man (y) & Wife (x,y)$\}$

where 'ϵ' is the null symbol.[6]

Second, note that the subject of a whether-question is a special case of the subject of a which-question (and so whether-questions are which-questions). Consider the examples of whether-questions given earlier. The subject of

Is John married?

is

$\{$married (John), $\sim$ married (John)$\}$

which can be put, as the subject of a which-question

$\{\epsilon$ //(x = John) & Married (x)$\}$

Again, the subject of

Has John stopped beating his wife?

can be put

$\{\epsilon$ //(x = John) & (stopped_beating_wife (x)$\}$

We will take the view, then, that whether-questions are but a special case of which-questions, and state our rules (governing the acceptability of answers) in terms of which-questions only.[7]

The request of a question is a four-tuple <Rrange, Rcomplete, Rdistinct, Rdefault>. The <u>selection-size specification</u> Rrange is a specification on the lower and upper limits of the size of the selection. The lower bound is some integer greater than or equal to one (any question requests at least an alternative be selected). The upper bound can be any finite positive integer greater than or equal to the lower bound, or may be ommited entirely (which we indicate by using the null symbol "ϵ"). Thus Rrange takes the form

$$\ell..u \; (1 \leq \ell \leq u)$$
or
$$\ell..\epsilon \; (1 \leq \ell).$$

The selection-size specification allows us to express the difference between

What is the only Indian restaurant in Topeka?

and

What are some Indian restaurants in Topeka?

The first question involves a request for a single alternative, so Rrange = 1..1, while the second question involves a request for at least one alternative but sets no upper limit on the answer, so Rrange = 1..ϵ.

The second element of the request is the completeness-claim specification Rcomplete. Some questions, like

Which Chinese restaurants in Topeka take credit cards?

implicitly require the respondent to make a claim that the list given is complete, while other questions, like

What are some Chinese restaurants in Topeka which take credit cards?

do not. The completeness-claim is always a comparison of the number of alternatives selected with the total number of true alternatives. The completeness-claim could be that all the true alternatives have been given, or all but one, or most, or 10%, but in practice only one completeness-claim is important, namely, the claim that all true alternatives have been given. This is an especially important claim if the database is incomplete. Accordingly, we will stipulate that Rcomplete take the form of either '$\forall$' or 'ϵ' depending upon whether the maximum completeness-claim (that all true instances have been selected) is specified, or else that no such claim is specified.

The distinctness-claim specification Rdistinct is a specification regarding whether or not the selection given contains any redundancies. For example, the statement

The Greenhouse and the Holiday Inn South restaurant are restaurants in Topeka.

would in most situations not be an acceptable answer to the question

What are two different restaurants in Topeka?

if it turns out that "The Greenhouse" is in fact just the name given to the restaurant in the Holiday Inn South.

As with completeness-claims, various sorts of distinctness-claims are theoretically possible (that some of the alternatives are distinct, that all but one are, and so on) but in practice only two are important. We will say Rdistinct has the form "$\neq$" or "ϵ", depending upon whether the answer is to include a claim that all the alternatives are distinct or whether no such claim is to be made. In most cases in ordinary contexts and with most databases, questions and queries assume that the selection given in response consists of entirely distinct individuals.

The final element of the request is the default specification Rdefault. Rdefault specifies what is to be done in the face of presupposition failure. A presupposition of a question is any statement that must be true if that question is to have any true direct answer. For example,

Has John stopped beating his wife?

takes as direct answers

John has stopped beating his wife.

and

John has not stopped beating his wife.

But for one of those alternatives to be true, it has to be true that he has beaten his wife.[8] Of course, this example is the classic example of a loaded question: whether John answers yes or not, he commits himself to the proposition that he has beaten his wife in the past. Questions that presuppose something false are called "loaded," and are best answered by a direct answer but a corrective answer (one that corrects false presupposition). Examples:

(question) Has John stopped beating his wife?

(direct answer) Yes.

(corrective answer) He has never beaten his wife.

(question) What is an Indian restaurant in Topeka?

(direct answer) The New Dehli Deli.

(corrective answer) There are no Indian restaurants in Topeka.

We will take the position that a person may well desire a corrective answer to his question if he cannot be given a direct answer--indeed, this seems to be the rule rather than the exception. We can allow the request for a correction in case of presupposition failure by indicating a default question D. (The guard question cannot, of course, be identical to the original question.) For example, the question

What is an Indian restaurant in Topeka?

can be guarded by the default question

Is there an Indian restaurant in Topeka?

The null symbol indicates that no default question is specified--we may suppose that an error message is printed in case of presupposition failure.

Here then are a few examples of questions and their formal expressions.

(1) Is Hunam a restaurant?

$?(1..1, \in, \in, \in) \{\in // (x = \text{Hunam}) \ \& \ \text{Restaurant}(x)\}$

or

$?(1..1, \in, \in, \in) \{\text{Restaurant}(x) // x = \text{Hunam}\}$

If "Restaurant" is a type.

(2) Which of these are (different) Chinese restaurants: Hunam, The Peking Duck, and Solley's?

$?(1..\in, \forall, \neq, \in) \{\text{Restaurant}(x) // \text{Chinese}(x) \ \& \ [(x = \text{Hunam}) \ v \ (x = \text{The\_Peking\_Duck}) \ v \ (x = \text{Solley's})]\}$

(3) Which if any of these are (different) Chinese restaurants: Hunam, The Peking Duck, and Solley's?

$?(1..\in, \forall, \neq, D) \{\text{Restaurant}(x) // \text{Chinese}(x) \ \& \ ((x = \text{Hunam}) \ v \ (x = \text{The\_Peking\_Duck}) \ v \ (x = \text{Solley's}))\}$

where

$D = ?(1..1, \in, \in, \in) \{(\text{Restaurant}(\text{Hunam}) \ \& \ \text{Chinese}(\text{Hunam})) \ v \ (\text{Restaurant}(\text{The\_Peking\_Duck}) \ \& \ \text{Chinese}(\text{The\_Peking\_Duck})) \ v \ (\text{Restaurant}(\text{Solley's}) \ \& \ (\text{Chinese}(\text{Solley's}))\}$

(4) What are two (different) Greek restaurants in Topeka?

$?(2..2, \in, \neq, \in) \{\text{Restaurant}(x) // \text{Greek}(x) \ \& \ \text{Located}(x, \text{Topeka})\}$

By making the exact form of the request and the subject explicit,we can formulate rules governing the form any acceptable answer to a given question must have. We will state these rules recursively. The first set of specific rules apply to questions without default questions specified. We then state a general rule that applies to any questions with a default question specified. In what follows, $\mathcal{K}$ will represent the question and α the answer.

First, consider a which-question that specifies a size range but specifies neither a completeness claim nor a distinctness claim. Then the constraint on any answer is that it be a conjunction of instances of the type constraints and matrix. Formally,

Rule 1 if $\mathcal{K}$ is

$?(\ell..u, \epsilon, \epsilon, \epsilon) \{ T_1x_1 ,,, \quad T_rx_r//Ax_1...x_n \}$

then α must be

$(T_1a_{1_1} \&...\&T_ra_{1_r} \& Aa_{1_1}...a_{1_n}) \&...\& (T_1a_{s_1} \&...\& T_ra_{s_r} \& Aa_{s_1}$ [illegible] $_n)$

$\ell \leq s \leq u$

Example:

What is an example of a friendly bear?

Rule 1 allows

Yogi is a bear and Yogi is friendly.

but does not allow

Yogi is not a bear and Yogi is friendly.

nor does it allow

Yogi is a bear and Yogi is not friendly.

Next, consider a which-question that specifies a selection size range and a completeness-claim. Then any acceptable answer must involve a conjunction where each conjunct satisfies the type and matrix conditions and where the number of conjuncts falls in the range indicated, and a claim that if any individuals satisfy the type and matrix conditions they are identical to the ones cited. Formally:

Rule 2 If $\mathcal{K}$ is of the form

$$?(\ell..u, \forall, \epsilon, \epsilon)\ \{T_1x_1, \ldots, T_rx_r // Ax_1 \ldots x_n\}$$

then α must be

$$\{[T_1a_{1_1} \& \ldots T_ra_{1_r} \ \&\ Aa_{1_1} \ldots a_{1_n} \ \& \ldots \&\ T_1a_{s_1} \& \ldots \&\ T_1a_{s_r} \ \&\ Aa_{s_1} \ldots a_{s_n}]$$

$$\& \forall x_1 \ldots \forall x_n \ \ [(T_1x_1 \ \& \ldots \&\ T_rx_r) \rightarrow (Ax_1 \ \ldots x_n \rightarrow ((s_{1,n} = a_{n_1,n})$$

$$\text{v} \ldots \text{v}\ (x_{1,n} = a_{s_1,n})))]\}$$

where $(x_{1,n} = a_{k_{1,n}})$ is the conjunction $(x_1 = a_{k_1})$ $\& \ldots \&$ $(x_n = a_{k_n})$ and $\ell \leq s \leq n$. We could also state the completeness-claim using type-restricted quantifiers:

$$(\forall x_1/T_1) \ldots (\forall x_r/T_r)\ (\forall x_{r+1}) \ldots (\forall x_n)\ [Ax_1 \ldots x_n \rightarrow ((x_{1,n} = a_{1_1,n}) \text{v} \ldots \text{v}$$

$$(x_{1,n} = a_{s_1,n}))]$$

Example:

Which bears are friendly?

Rule 1 sanctions

Yogi is a bear and Yogi is friendly and for any x, if x is a bear then if x is friendly then x is identical to Yogi.

but does not sanction

Yogi is a bear and Yogi is friendly.

Next, consider a which-question that specifies a selection-size range and a distinctness-claim, but no completeness-claim. Then an acceptable answer will assert a conjunction where each conjunct satisfies the type and matrix conditions and where the number of conjuncts falls in the specified range and where an assertion is made that no two individuals cited are the same. Formally:

Rule 3: If $\mathcal{K}$ has the form

$$?(\ell..u, \in, \neq, \in)\left\{T_1x_1,\ldots, T_rx_r//A_{x_1}\ldots x_n\right\}$$

then α must have the form

$$\left\{[T_1a_{1_1}\&\ldots\&T_ra_{1_r}\ \&\ Aa_{1_1}\ldots a_{1_n}\&\ldots\&\ T_{1}a_{s_1}\ \&\ldots\&\ T_{1}a_{s_r}\ \&Aa_{s_1}\ldots a_{s_n}]\right.$$
$$\left.\&\ [\&_{(1\leq i\leq j\leq p)}V_{(1\leq k\leq n)}\ (a_{i_k}\neq a_{j_k})]\right\}$$

Where $\ell\leq s\leq u$ and where $V_{(a\leq k\leq n)}\ (a_{i_k}\neq a_{j_k})$ abbreviates $(a_{i_1}\neq a_{j_1})\ v\ldots v\ (a_{i_n}\neq a_{j_n})$, i.e., the statement that the ith and jth conjuncts are distinct, and so the full claim that all conjuncts are distinct will look like $\&\ (1\leq i\leq j\leq p)\ V\ (1\leq k\leq n)\ (a_{i_k}\neq a_{j_k})$. So, for example, to say that the list (11, 13, 17) is not redundant is to say (11 ≠ 13) and (11 ≠ 17) and (13 ≠ 17). Also, to say that the list of pairs ((1,2),(1,3),(3,4)) is not redundant is just to say {[(1 ≠ 1) v (2 ≠ 3)] & [(1 ≠ 3) v (2 ≠ 4)] & [(1 ≠ 3) v (3 ≠ 4)]}.

Example:

Which different bears are friendly?

Rule 3 accepts

Yogi is a bear and Yogi is friendly and Winnie is a bear and Winnie is friendly and Yogi is not identical to Winnie.

but does not accept

Yogi is a bear and Yogi is friendly and Winnie is a bear and Winnie is friendly.

Finally, consider a which-question that specifies a size range, completeness-claim, and distinctness-claim. Then the form of the answer is just the fusion of the two rules above.

Rule 4: If K has the form

$?(\ell..u, \forall, \neq, \epsilon) \quad T_1x_1,\ldots,T_rx_r//Ax_1\ldots x_n$

then α must be of the form

$$\{[T_1a_{1_1} \&\ldots\& T_ra_{1_r} \& Aa_{1_1}\ldots a_{1_n} \&\ldots\& T_1a_{s_1} \&\ldots\& T_ra_{s_r} \& Aa_{s_1}\ldots a_{s_n}]$$

$$\&(\forall x_1)\cdots(\forall x_n) \; [(T_1x_1 \&\ldots\& T_rx_r) \rightarrow (A_{x_1}\ldots x_n \rightarrow ((x_{1,n}=A_{j_1,n}) \vee\ldots\vee$$

$$(x_{1,n}=a_{s,n})))] \;\&\; [\mathop{\&}(1 \leq i \leq j \leq p) \bigvee (1 \leq k \leq n) \quad (a_{i_k} \neq a_{j_k})]\},$$

$\ell \leq s \leq u.$

Example: Which different bears are friendly?

Rule 4 sanctions:

Yogi is a bear and Yogi is friendly and Winnie is a bear and Winnie is friendly <u>and</u> for any x, if x is a bear then if x is friendly then either x is identical to Yogi or x is identical to Winnie, and Yogi is not identical to Winnie.

So much for questions that do not involve a default specification. Questions that do specify a default question can be handled easily.

Rule 5: Let K_δ be a question involving a specified default question δ and let K_ϵ be the same question without a default specification. Then any acceptable answer α to K_δ will have the form α_1 or α_2, where α_1 is an acceptable answer to K_ϵ and α_2 is an acceptable answer to δ.

Example:

What (if any) is an Indian restaurant in Topeka?

Rule 5 allows

The New Dehli Deli

and

There are no Indian restaurants in Topeka

but not

<error>

The grammar of elementary questions, then, consists of FOL plus a new operator"?," a few obvious additional rules of construction (for the various particular forms of "?<Rrange, Rcomplete, Rdistinct, Rdefault> S") and five rules governing the acceptability of answers. These latter rules are the heart of the subject, of course; they are perhaps reminiscent of rule of inference or transformation in FOL itself, but, of course, are not to be considered as such. They are better called "rules of response," rules which govern, not how information is to be transformed, but how (in what form) it is to be conveyed.

3) Formal Queries

We now have an apparatus for specifying questions exactly. This apparatus allows us to get a start on the formal specification of the correctness of answers. An answer to a given question is correct if and only if: (a) it has acceptable form and (b) it is true. And we can now specify (by rules 1-5) what it is for an answer to have an acceptable form. We now want to specify formally what it is for an answer to be true.

Here databases enter the picture. We take the view that answers are true or false only against the backdrop of some body of information, some database. And so if we are to develop some formal account of truth relative to a database, we need to develop some formal model of databases.

Reiter [15] contrasts two ways to formally represent databases. The first he calls the "model theoretic" view, the dominant view since the work of Codd and others in the early 70's. The second he calls the "proof theoretic" view. Reiter argues that the proof theoretic view is better in that it naturally allows for the representation of disjunctive information, null values, general facts, events, property inheritance (IS-A) hierarchies, and other "world knowledge." I find his arguments persuasive, but will not rehearse them here. Rather, I will work with the proof-theoretic model because it very nicely compliments the erotetic apparatus outlined earlier.

Under the proof theoretic view, a data base is taken to be a first order theory (in an underlying first order relational language). A first order theory $\mathcal{T}$ of language L is just a subset of the well formed formulas of L. For example, with FOL as defined earlier, we might have a first order theory τ with ground atomic formulas:

$\tau = \{$Restaurant (Hunam), Restaurant (The_Peking_Duck), Restaurant (The_New_Dehli_Deli), Chinese (Huna,), Chinese (The_Peking_Duck), Indian (The_New_Dehli_Deli), Located (Huna, Topeka), Located (The_Peking_Duck, Topeka), Located (The_New_Dehli_Deli, Topeka), Hunam=Hunam, The_New_Dehli_Deli=The_New_Dehli_Deli, The_Peking_Duck =The_Peking_Duck, Topeka = Topeka$\}$

We can then express truth with respect to the database in terms of derivability:

$$\tau \vdash \alpha$$

means that α is derivable from τ taken as a set of premises. In our example above, clearly

$\tau \vdash$Restaurant (Hunam)

i.e., the fact that Hunam is a restaurant is derivable from the database--trivially, since it is an explicit member of the set. Less trivially,

$\tau \vdash (\exists x)$ (Chinese(x) & Restaurant (x) & Located (x, Topeka))

i.e., we can derive from the theory that there is at least one Chinese restaurant in Topeka. We can now try a plausible approach to defining what it is for an answer α to a query K applied to a database τ to be true: α is true with respect to K if and only if $\tau \vdash \alpha$. More generally, a given question K applied to a database τ is <u>correctly answered</u> by α if and only if (a) α is of acceptable form as an answer to K (i.e., meets rules 1-5) and (b) $\tau \vdash \alpha$.

Let's follow this suggestion to see where it leads. Suppose we ask (of our sample database)

Which restaurants in Topeka are Indian restaurants?

The correct answer should be

(Restaurant (The_New_Dehli_Deli) & Indian (The_New_Dehli_Deli) & Located (The_New_Dehli_Deli, Topeka)) & (∀x) [(Restaurant (x) & Indian (x) & Located (x, Topeka)) → (x = The_New_Dehli_Deli)]

But while $\tau \vdash$ (Restaurant (The_New_Dehli_Deli) & Indian (The_New_Dehli_Deli) & Located (The_New_Dehli_Deli, Topeka)), we cannot derive the completeness-claim (∀x) [(Restaurant (x) & IndiaN (x) & Located (x, Topeka)) → (x = The_New_Dehli_Deli)]. Since universal claims cannot be derived from a theory consisting solely of ground atomic formulas, Reiter suggests first adding to τ a <u>domain closure axiom</u> to the effect that all individuals in the interpretation are named in the database. In the example above, this axiom has the form:

(∀x) [(x = Hunam) v (x = The_Peking_Duck) v (x = The_New_Dehli_Deli) v (x = Topeka)]

More generally, if the domain of the interpretation is $(d_1,\ldots,d_n)$ the domain closure axiom is:

$(\forall x)\ ((x = d_1) \vee \ldots \vee (x = d_n))$

Also, we must add to completion axioms for each predicate (to allow the derivability of negations such as Restaurant (Topeka). In our small example:

(∀x) [Restaurant (x) → ((x = Hunam) v (x = The_Peking_Duck) v (x = The_New_Dehli_Deli)]

(∀x) [Indian (x) → (x = The_New_Dehli_Deli)]

(∀x) [Chinese (x) → ((x = Hunam) v (x = The_Peking_Duck))]

(∀x) (∀y) [Located (x,y) → (((x → Hunam) v (x = The_New_Dehli_Deli)v (x = The_Peking_Duck) & (y = Topeka)))]

Happily, this all accords with out intuitions: to correctly answer a question that demands a completeness-claim, we have to know that the database to which the question is addressed is complete.

Again, suppose that we ask of our sample database

What are two different Chinese restaurants in Topeka?

The correct answer should be

(Chinese (Hunam) & Restaurant (Hunam) & Located (Hunam, Topeka)) & (Chinese (The_Peking_Duck) & Restaurant (The_Peking_Duck) & Located (The_Peking_Duck, Topeka)) & ~(Hunam = The_Peking_Duck)

But from , even including the closure and completion axioms, we cannot derive the distinctness-claim

~(Hunam = The_Peking_Duck)

We need to add more axioms to , namely, <u>unique name axioms</u>

$\sim(C_i = C_j)$

for all (C_i, C_j), $i \neq j$, in the set of constants of our relational language. When those axioms <u>are</u> added, the distinctness claim is derivable.

But again this accords well with our intuitions. To answer correctly a question that demands a distinctness claim, we have to know that our database contains no duplicate names. This is standardly assumed about databases in practice, and it is a very nice result that our formal specification mechanism forces us to make that assumption explicit.

When $\mathcal{T}$ consists of the ground atomic formulas, domain closure axiom, unique name axioms, completion axioms (and equality axioms specifying the reflexibility, transitivity and symmetry of the equality predicate), the only model for $\mathcal{T}$ is the original interpretation, and thus truth in the interpretation amounts to provability in the theory. This completes our definition of correctness of answers to queries.

4) Conclusion

I have argued above that a relatively unnoticed area of fourth-generation logic, erotetic logic, devised by philosophical logicians over the last thirty years or so, can be used to complement database theory. In particular, erotetic logic complements very nicely the approach to representing databases taken by Reiter and other workers in the area of the domain calculus, enabling us to formally represent questions and the correctness of answers.

Of course, I suspect that there are other benefits to employing erotetic logic in database theory. One area of possible interest is the verification of relational database queries. That is, given a question posed in natural language, we may be able to verify that a query posed in one of the relational database query languages (domain algebra, domain calculus, transform languages such as SEQUEL/SQL, or graphic systems such as Query-By-Example) will correctly answer the question. For example, to verify that

Which Chinese restaurants in Topeka take credit cards?

will be correctly answered by the SEQUEL query

```
SELECT NAME
FROM RESTAURANT
WHERE TYPE='CHINESE'
   AND CREDIT = 'YES'
   AND LOCATION = 'TOPEKA'
```

applied to a database containing the relation

RESTAURANT (NAME, LOCATION, TYPE, CREDIT)

we need to verify that

$?(1..\epsilon, \forall, \epsilon, \epsilon)$ $\{$Restaurant(x)//Chinese(x) & Located (x, Topeka) & takes_credit_cards(x)$\}$

will be correctly answered by the query commands. Assuming that the implementation of the select command has been verified (it is presumably just a linear search routine, which can be verified in the standard way), we can show that the query will produce a complete list of names that fit the matrix and type constraints. However, we still need to conjoin the completeness-claim explicitly since it is not present as a datum in the relational database. The point here is that Reiter's view of a database as a first order theory is not something we can immediately implement in a standard relational database. A suggestion worth pursuing is that Reiter's view be implemented in PROLOG. After all, PROLOG does allow the inclusion of general rules as well as particular facts in the database.

In addition, AI researchers may find erotetic logic useful. Generally, erotetic logic gives us a handle on an important aspect of natural language, viz., the asking and answering of questions. More specifically, a number of philosophers of science have pointed out the central role why-questions play in scientific research (see esp. [20]). Besides "capturing" (i.e., representing) world knowledge, we need to understand the various and intricate ways people interrogate that knowledge. Erotetic logic can help us in that quest.

NOTES

[1]Logicians vary in their enthusiasm for fourth generation logic, from unbridled enthusiasm to skepticism (even outright hostility). Typical of the enthusiasts is Routley:

> On the whole there has been far too much effort expended on trying to accommodate philosophical clarifications to going logical systems and strait jackets...rather than trying to develop logical systems to handle the evident data and to deal with going philosophical problems. Classical logic (i.e., FOL), although once and briefly an instrument of liberation and clarification in philosophy and mathematics, has, in becoming entrenched, become rigid, resistant to change and highly conservative, and so has become an oppressive and stultifying influence.... Classical logic is, as now enforced, a reactionary doctrine.
>
> Routley quoted in Brodie, et. al. 1984, p. 143

Routley touches upon one of the main arguments employed by many fourth generation logicians: that logic ought to be "empirical," that is, should square with the facts of cognitive life. And fourth generation logicians argue that FOL is not empirically adequate.

Many logicians are skeptical, however. Quine ([13]p. 86) is only one of many who resist the move to go beyond FOL:

> ...let us not underestimate the price of a deviant logic. There is a serious loss of simplicity... And there is a loss, still more serious, on the score of familiarity.... The price is perhaps not quite prohibitive, but the returns had better be good.

Those who are reluctant to embrace fourth generation logic seem to me to have two basic objects: first, that the newer logics are not extensional (hence lack many nice features present in FOL, which is extensionalist); second, that the newer logics are proliferating, yet don't fit together well (hence lack the elegance and simplicity of FOL). Let me elaborate.

We say that a logical expression is extensional if and only if coreferential expressions can be substituted without changing the truth value. (That means replacing singular terms with terms of the same denotation (e.g., 'Mark Twain' and 'Sam Clemens'), predicates with predicates of the same extension (e.g., 'man' and 'featherless biped'), or sentences with sentences of the same truth value.) While FOL is extensional, fourth generation assertoric logics are typically intensional (i.e., non-extensional). Thus they often lack the simple decision procedures found in classical propositional logic.

And again, fourth generation logics are bewildering in their variety. Some seem to be more extensions of FOL than replacements, others seem to be wholesale replacements. It is quite unclear whether they all can be fitted together, and if so, just how.

[2]The reader may wish to see Belnap and Steele's discussion of compound questions [5] pp. 87-107.

[3]This system is similar to that of Belnap and Steele [5] which grew out of Belnap's earlier work [4] with a number of simplifications and modifications. In particular, I view whether-questions as special cases of which-questions and I build default questions (guards) into the specification of simple questions. I do this to being the symbolism closer to actual database queries. (The reader may wish to read the alternate system devised by Aquist [2]. For an elegant and highly formal account, see Kubinski [11].)

[4]For a review of FOL, see [12] or [17]. In particular, the notion of interpreting a first order language is discusses in Chapter Two of

[12] and Chapter Two of [17].

[5]Types are discusses in [17] pp. 88-92; and also in [14]. Note that instead of using types, one can use a many-sorted logic to the same effect. For an explanation of the role of category or type conditions in erotetic logic, see Belnap [4] pp. 72-73.

[6]We can do the converse, i.e., push matrix conjuncts over into the type constraints, only if those conjuncts are types or if we are willing to allow new types to be created at will-not a very practical approach.

[7]This formal reduction can have a cost in simplicity. For example, the subject of the whether-question

Which of the following are on sale: ham, chicken, steak, lobster, and turkey?

is

$\{$(on_sale (ham, today), on_sale (chicken, today, on_sale (steak, today), on_sale (lobster, today), on_sale (turkey, today))$\}$

but turned into the subject of a whether-question is

$\{\epsilon$ //on_sale (x, today) & ((x = ham) v (x = chicken) v (x = steak) v (x = lobster) v (x = turkey))$\}$

which is rather more awkward.

Even more awkward are questions involving quantified statements, e.g. the subject of

Are all swans white?

is

$\{(\forall x)\ (\text{swan}(x) \rightarrow \text{white}\ (x)\ ,\ \sim (\forall x)\ (\text{swan}(x) \rightarrow \text{white}(x)\}$

That can only be represented as the subject of a which-question if we use a universal type U_x true of all individuals (or if dealing with a language with no types, the function Axv $\sim$ Ax will serve as well). Then the equivalent subject is

$\{\epsilon // \mathcal{U}\, x \,\&\, (\forall x)\ (\text{swan}(x) \rightarrow \text{white}(x))\}$

[8]This point is more subtle if we interpret the question as a whether-question:

$\{\epsilon\ //\ (x = \text{John})\ \&\ \text{has\_beaten\_wife (John)}\ \&\ \text{now\_beats\_wife (John)}\}$

Posing the question with that subject gets as direct answers

Has_beaten_wife (John) & now_beats_wife (John)

and

$\sim$[has_beaten_wife (John) & now_beats_wife (John)]

That latter alternative does not logically imply that John did beat his wife. However, it is still true under this construal that to answer "Has John stopped beating his wife?" by "no, it is not the case that he has stopped beating his wife" does not correct the presupposition of the question.

BIBLIOGRAPHY

[1] Anderson, Alan Ross and Nuel Belnap (1975) Entailment: The Logic of Relevance and Necessity (vol. I) Princeton: Princeton University Press.

[2] Aquist, Lennart (1965) A New Approach to the Logical Theory of Interrogatives. Uppsala: Filosofiska Studier.

[3] Bartley, W. W. (1977) Lewis Carroll's Symbolic Logic NY: Clarkson Potter, Inc.

[4] Belnap, Nuel D. (1963) An Analysis of Questions: Preliminary Report. Technical Memorandum issued by System Development Corporation.

[5] Belnap, Nuel and Thomas Steel (1976) The Logic of Questions and Answers. New Haven: Yale University Press.

[6] Bridge, Jane (1977) Beginning Model Theory Oxford: Oxford University

[7] Bromberger, Sylvain "Why-Questions" in Mind and Cosmos ed. by Robert Colodny (1966) University of Pittsburgh Press.

[8] Haack, Susan (1978) Philosophy of Logics Cambridge: Cambridge University Press.

[9] Hughes, G. E. and M. J. Cresswell (1968) An Introduction to Modal Logic London: Methuen.

[10] Hughes, G. E. and M. J. Cresswell (1984) A Companion to Modal Logic London: Methuen.

[11] Jørgensen, Jørgen (1931) A Treatise of Formal Logic Copenhagen.

[12[Kubinski, Tadeusz (1980) An Outline of the Logical Theory of Questions

[13] Mendelson, Elliot (1979) Introduction to Mathematical Logic (2nd Ed) NY: Van Nostrand (Chapter 2)

[14] Quine, W. V. (1970) Philosophy of Logic Englewood Cliffs, N.J.: Prentice-Hall

[15] Reiter, Raymond (1984) "Towards a Logical Reconstruction of Relational Database Theory" in Michael Brodie et. al. (eds) On Conceptual Modeling NY: Springer-Verlag.

[16] Rescher, N. (1969) Many-Valued Logic NY: McGraw-Hill.

[17] Rescher, Nicholas and Urquhart, A. (1971) Temporal Logic NY: Springer Verlag.

[18] Shoenfield, Joseph (1967) Mathematical Logic Reading, Mass: Addison-Wesley (Chapter 2).

[19] Thomas, Stephen (1986) Practical Reasoning in Natural Language (3rd Ed.) Englewood Cliffs, N.J.: Prentice-Hall.

[20] Turner, Raymond (1985) Logics for Artificial Intelligence Chichester: Ellis Horwood Ltd.

EROTETIC LOGIC AS A SPECIFICATION LANGUAGE FOR DATABASE QUERIES

by

GARY JAMES JASON

Ph.D., University of Illinois, 1982

AN ABSTRACT OF A MASTER'S THESIS

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1986

AN ABSTRACT OF EROTETIC LOGIC AS A
SPECIFICATION LANGUAGE FOR DATABASE QUERIES

Several nonstandard logics (including many-valued, temporal and fuzzy logics) have been usefully applied by computer scientists. In this paper, erotetic logic--the logic of questions--is reviewed, and its use in the specification and verification of database queries is explored. Erotetic logic is classified as a nonassertoric fourth generation logic in §1 of the paper. In §2, a formal erotetic logic is described. In §3, the logic is explored as a tool for precisely stating database queries and for specifying the correctness of answers. In §4, the possibility of using the erotetic system for verifying query functions is briefly discussed, and some general conclusions drawn.